10/525381
DT01 Rec'd PCT/PTC 23 FEB 2005

MEDIA ARTICLE COMPOSITION

The present invention relates to a method of, and apparatus for, composing a media article.

Media articles portray content (whether real, imagined or computer-generated) to a person's senses. Media articles can be presented to a person via a variety of media, including text, voice, sound, pictures or moving images.

As recording technologies have improved, the amount of recorded media articles available to a consumer has grown rapidly. Media articles are often recorded in media files (note that although the plural 'media' is used here, 'media file' is to be understood to include both files which are intended to be conveyed to user by only one medium – e.g. text or speech and also 'multimedia' files whose meaning is conveyed by a plurality of media). The Internet is the most recent communications network to emerge and provides worldwide transmission of recorded digital media files representing text, sound, pictures, moving images or a combination of these. Since the number of media files accessible via the Internet is so large, there is a need to label media files with some description of what they contain. Thus, for example, HTML (HyperText Mark-up Language) files contain 'meta' tags which include keywords which indicate what subjects are covered in the web-page presented to the user.

Labelling media files with metadata is made more beneficial when a group of users agree on how that metadata should be structured and the elements it should contain. Often, XML (eXtensible Mark-up Language) is used to define such structure and the elements contained within that structure. In effect, XML can be used to define metadata 'languages'. One example of such a metadata 'language' is StoryML, as discussed in "StoryML: An XML Extension for Woven Stories" by P. Gerdt et al, pp893 to 902 of the proceedings of the Intelligent Tutoring Systems conference 2002. StoryML is a metadata language designed to describe a contribution to a collaboratively written story. As such it includes elements giving the author of the contribution and the contributions relationship to other contributions.

A proposal for adding metadata to video files (the 'Multimedia Content Description Interface' more widely known as MPEG-7) is being discussed by the Moving Pictures Expert Group.

International Patent application WO 02/057959 discloses computer software providing a user with a tool to organise media files. Various metadata can be associated with those files. The files and the metadata are stored in a 'relational' database - note that 'relational' as used in the expression relational database has little to do with relations between the database entries or what is represented in those entries - instead it refers to a 'relation' in the sense that word is used in mathematical set theory.

One method of composing a media article involves the putting together of a plurality of components. For example, a film is made up of a plurality of scenes as set out in a screenplay. There have been some attempts to create a media article automatically in this way. For example, the production of a Waltz in accordance with a musical dice game is found at http://sunsite.univie.ac.at/Mozart/dice/ . Similarly, an automatic story generation program, the Romance Writer is available from at http://familygames.com/features/humor/romance.html .

According to a first aspect of the present invention, there is provided a method of automatically composing a media article comprising:

analysing digital metadata associated with a first set of stored media data, which digital metadata includes:

related set identity data identifying a second set of stored media data; and

relationship data which indicates the relationship between what is represented by the first set of stored media data and what is represented by the second set of stored media data; and

arranging said first and second sets of stored media data in a media article in accordance with said analysis.

By analysing digital data associated with a first set of stored media data, which digital data includes an identifier of a second set of stored media data and an indication of the relationship between what is represented by the first set and what is represented by the second set, and arranging the first and second sets of stored media data in a media article in accordance with said analysis, a method of composing a media article is provided which obviates the need for another source of sequencing information to be provided at the time the media article is composed.

The expression 'set of stored media data' includes media data files, streams, or a set of pointers into a file or database.

Digital sets of stored media data are stored in a variety of formats. The expression set of stored media data is not intended to be limited to any particular format and so includes, for example, a file which merely contains data on the position of components seen by a user when playing a computer game – the data in that file subsequently being processed by rendering software to generate an image for display to the user.

In some embodiments, said method further comprises generating said set identity data and said relationship data.

Preferably, said metadata further comprises content data indicating what is represented by said sets of stored media data, said method further comprising the step of selecting, from a plurality of sets of stored media data, one or more sets of stored media data in dependence upon said content data, said one or more sets including said first and second sets of stored media data.

This combination of a) searching to provide a plurality of potential components for a media article, or part of a media article; and b) subsequently arranging a plurality of the selected components in accordance with metadata associated with those components allows the automatic composition of media articles directed towards a particular theme more easily than has hitherto been possible.

Preferably, the method further comprises making a plurality of such selections; and concatenating the results of said selections.

This allows an approach to the composition of a media article which broadly follows an established pattern, but which allows a degree of flexibility within that pattern not seen in conventional systems. Thus the type of broad pattern known for narratives and films – for example, the StoryCraft program from StoryCraft Corporation, 560 Roland Drive, Norfolk, VA 23509, USA helps an author write a story by asking the writer to introduce the hero and an antagonist, prior to some conflict between them, which is then followed by the hero's triumphal return home – can still be used, but variations within that pattern can easily be introduced by changing the nature of the selection.

This will allow, for example, the cost effective creation of different versions of a film or computer game, conforming to different artistic or qualitative ambitions, whilst still retaining subjectively high standards of narrative, motion and audio continuity.

According to a second aspect of the present invention, there is provided a media article composition apparatus comprising:

one or more memory devices storing, for each of a plurality of sets of stored media data, metadata including relationship data indicating one or more relationships between the content represented in said set of stored media data and the content represented in one or more other sets of stored media data; and

one or more processors in communication with said one or more memory devices and arranged in operation to compose a media article by arranging said sets of stored media data or identifiers thereof in accordance with said relationship data.

In preferred embodiments, an object-oriented database is used to store objects containing metadata associated with a set of stored media data identified by said metadata. Relationship metadata can then be represented by relationships between objects in the object-oriented database.

The sets of media data might alternatively be stored in a file system.

By way of example only, specific embodiments of the present invention will now be described with reference to the accompanying Figures in which:

Figure 1 is a schematic illustration of components of a personal computer which may be operated to provide a media assembly system in accordance with a first embodiment of the present invention;

Figure 2 is a diagram of the architecture of a computer program used in controlling the personal computer of Figure 1 in accordance with a first embodiment of the present invention;

Figure 3 shows metadata associated with a media element following the use of the media mark-up tool of Figure 2;

Figure 3B shows a cause object stored in the object-oriented database of Figure 2;

Figures 4A to 4G show how an editor builds the relationship data included in the metadata of Figure 3;

Figure 5 is a hierarchical representation of the relationship data entered by the editor as shown in Figures 4A to 4H;

Figure 6 shows template data generated using the template creation tool of Figure 2;

Figure 7 shows user profile data generated using the user profile creation tool of Figure 2;

Figure 8 is a flow chart showing the operation of the template populator component of the first embodiment of the present invention;

Figure 9 shows the operation of the template populator in relation to the first section of the template illustrated in Figure 7;

Figures 10A to 10D show the operation of the template populator in relation to the second section of the template illustrated in Figure 7;

Figure 11 shows an edit decision list as might be produced by the template populator module;

Figure 12 shows the display presented to the user on executing a computer program which provides a preferred embodiment of the present invention;

Figure 13 shows the same display at a later stage of operation of the program;

Figure 14 shows an additional sub-window which opens on the user selecting one of the media bins;

Figure 15 shows an additional sub-window presented to the user when he selects one of the media elements contained within the selected media bin;

Figure 16 shows the application window after a user indicates that he wishes to see the relationship data associated with selected (or all) media elements contained within a media bin;

Figure 17 shows an additional sub-window which enables the user to define vocabularies;

Figure 18 shows the display after the user has selected a template for use in generating a media article, and is editing that template;

Figure 19 shows the display after the user has selected a template for use in generating a media article, and is expanding that template;

Figure 20 shows a sub-window which is displayed to show the results of the user running the template populator component of the program;

Figure 21 illustrates additional functionality provided to a user via a query builder window (as seen in Figures 18 and 19) in a refinement of the preferred embodiment of the present invention; and

Figure 22 shows additional controls provided to a user when previewing the media article using the refinement of the preferred embodiment of the present invention.

Figure 1 shows a personal computer which comprises well-known hardware components connected together in a conventional manner. The well-known hardware components comprise a central processing unit 10, random access memory 12, read-only memory 14, a hard disk 16 and input/output devices 18,20,22,24, and 26. The hardware components are interconnected via one or more data and address buses 28. The input/output devices comprise a monitor 18, a keyboard 20, a mouse 22, a CD ROM drive 24 and a network card 26. The network card is connected to a server computer 30 by the public Internet 32.

Figure 2 shows the software and data components of a system according to a first embodiment of the present invention. The software includes three input program modules, namely a media mark-up tool program 40, a template creation tool program 42, a user profile creation tool program 44, object-oriented database management software 45, and two output program modules, namely a template populator program module 46 and a content synthesis program module 48.

The system also includes two stores for the persistent storage of data. The first of these is a content store 50 which comprises a number of media files stored on the hard disk 16 using the file system provided by the computer's operating system program. The second store for the persistent storage of data comprises an object-oriented database 54 known as ObjectStore® supplied by Excelon Corporation, Burlington, Massachusetts. The database stores three different categories of objects, namely media objects 51 (each of which includes metadata describing one of the media files stored in the content store 50), template objects 52 and user profile objects 53. The objects in the database are again stored on the hard disk 16 of the computer. Persistent storage for the object-oriented database and/or the content store might, instead of the hard disk, be provided by removable data carriers such as DVDs, CDs, CD-ROMs or on different computers (accessed for example via a URI) accessible via a network connection such as is provided by the network card 26.

The three input program modules control the computer (Figure 1) to provide a user interface allowing the generation of objects which include data structures constructed in accordance with predetermined structured data models. This increases the ease with which different program modules can exchange data. Where the structured data model is made public, different parties can write different components of the program which are nevertheless able to exchange data with one another. Also, objects created by one editor can be used by another editor working at a different location or at a later time.

In the present embodiment, the three input program modules offer the following functionality:

The media mark-up tool provides an interface for a editor to update the content store 50 and the object-oriented database 54. In practice it is envisaged that an editor using the present embodiment will have access to media elements generated by other editors, rushes from various sources, sections of prepared programmes, still photographs and various other pieces of media (all represented in electronic form) at his disposal. These media elements are stored in an appropriate directory structure in the content store. Each directory is known as a 'bin' in the art – a reference to the labelled bins in which rolls of film relating to a particular project are stored.

Media elements are often recorded digitally in a file which may contain a number of elements in a linear stream. Such a file may therefore contain one or more media elements associated with it. The media mark-up tool allows the editor to preview files and, where there are to be several media elements within that file, set start and end points defining the scope of each media element within the file. If there is only one media element associated with the file then the start point is simply zero and the end point is the length (duration) of the media. In this way, a editor is able to generate a plurality of files, each forming one media element. The editor gives a name to each file at the time he defines the start and end points of the media element.

However, for the purposes of the present description, it is assumed that the editor begins only with a file that includes an electronic representation of unedited film recorded at a football match and introduction sequences for a football programme etc. An unedited piece of film is known as a 'rush' in the art. Using the media mark-up tool 40, the editor might select various sections of the rush and store each as a media element in a shorter file in a directory in the content store 50.

The media mark-up tool also provides a tool enabling the editor to generate or edit metadata for media elements stored in the content store 50. The tool stores this metadata as an object in the object-oriented database 54.

On selecting a directory within the content store, the editor is provided with a graphical user interface which presents set of icons (Figure 4A), each of which represents one of the media elements held in that directory. These icons may just be a still from a video sequence contained within the file, for example. In the examples below, they are illustrated as boxes which include the media element identifier entered by the editor in the top right-hand corner and which contain a description of what occurs in the media element in the centre of the box. The user interface allows the editor to select the media elements using the mouse 22 and to move the media elements around the screen using the mouse.

Having selected one of the media elements, the editor enters metadata to be associated with that media element in two stages. In a first stage, the editor can double-click on one of the pictures to bring up a form onto which the values of the parameters included within the schema can be entered.

An example of the metadata generated in the first stage is shown in the second to twelfth row of Figure 3 (the information in the first row having been generated when the editor gave a media element identifier to the file).

It will be realised that the metadata is arranged in accordance with a structured data model. In each row, the entry at the rightmost column represents the value of a property which is input by the user. The structured data model may provide that a plurality of properties should be labelled as members of a unit at a first level of aggregation – here referred to as a set of properties (column second from the left in those rows which have four columns). The structured data model may also provide that a plurality of sets should be labelled as members of a unit at a second level of aggregation – here referred to as a superset of properties (leftmost column in those rows which have three or four columns). Those skilled in the art will realise that further levels of aggregation might be provided.

The hierarchical arrangement is influenced by the Multimedia Content Description Interface mentioned above. The intention is not to enforce usage of a complete data model across all possible applications, but to enable re-use of content within the subject domain of a production company or a specific set of projects (eg. wildlife documentaries). The data model provided is intended to provide a maximal set of elements and an interface which assists their use and the vocabularies which can be applied to them.

The metadata includes a variable number of parameters (but must nevertheless conform with the predetermined structured data model). In the example, shown in Figure 3, the editor has entered values for 18 properties. These include:

i) Media Element ID – this identifies the media element – in the present example, the editor has given it a numerical value of 0.xx, where xx reflects the position of the media element within the original rush;

This is followed by a 'Media' superset which comprises two properties and a 'Position' set of properties. The two properties are:

ii) URI – the Universal Resource Identifier of the file which contains the media element;

iii) Format – this gives an indication of the format of the data making up the file;

The 'Position' set contains two properties as follows:

iv) In – an indication of the time elapsed since the start of the rush at the start of the media element;

v) Out - an indication of the time elapsed since the start of the rush at the start of the media element;

The 'Media' superset is followed by a superset of four 'structural' properties. That superset begins with

vi) Description – a description of the content of the file;

which are followed by another set (called 'Event') which contains three properties:

vii) Nature – the type of event that is seen in the video sequence recorded in this file;

viii) Performer – the person performing the principal action of the event;

ix) Recipient – the person subject to the principal action of the event;

These properties are followed by a domain-specific superset of properties which are only sensibly applied to media elements which relate to material obtained from two-sided sporting events;

The first two properties belong to a set (called 'Teams') of two properties:

x) Home Team – the name of the team playing on their home ground during the football match featured in the original rush;

xi) Away Team - the name of the other football team in the football match featured in the original rush;

This set is followed by the two properties:

xii) Performer Allegiance – the side (if any) to which the performer owes allegiance;

xiii) Recipient Allegiance - the side (if any) to which the recipient owes allegiance;

These two properties are followed by a set (named 'conceptual') containing two properties:

xiv) Interest Value – this value, between 0 and 1 indicates how significant the editor considers this media element to be; and

xv) Rating – this value indicates the suitability of the media element for showing to people based on an age criterion - in a similar way to the classification given to films.

Once the editor has entered this data, the picture is replaced with the description of the media element given as the value of the 'description' property above. The form is then removed from the display to return to the display of a plurality of pictures representing the media elements selected by the editor from the original rush (Figure 4A).

The media element metadata is then stored as a media object in the object-oriented database 54.

The second stage of the metadata creation which generates one or more 'Relationship' properties will now be described in relation to Figures 4A to 4H. The user is provided with a graphical user interface, allowing him to indicate relationship properties between media elements by moving and clicking on icons representing those media elements on the screen 18 of the PC (Figure 1).

One relationship which the editor may indicate is a causal relationship. To do this, the editor clicks on a button element presented on the screen (not shown), which changes the form of the cursor. Thereafter the user moves the cursor to an media element which he judges to be a cause of another media element. He then clicks on the media element to identify it as a causal media element and moves the cursor to the media element which represents the effect of that cause and clicks again. Once he has done this, an arrow is drawn from the first media element to a diamond shape representing a cause object and then a second arrow is drawn from the diamond to a second media element. An editor may wish to make a causal association of this type when he considers that a viewer seeing the media element representing the effect would also wish to see the cause. In the example shown in Figure 4B, the editor has indicated that:

i) media element 0.13 is caused by media element 0.12, and that both media element 0.14 and 0.15 are caused by media element 0.13; and

ii) media element 0.53 is caused by media element 0.52 which is in turn caused by media element 0.51.

In response to the input of a causal association, a cause object (Figure 3B) having its own media element identifier is added to the object-oriented database 54. The metadata associated with the media element representing the effect is updated to include a parameter indicating the media element identifier of the cause object. In more detail, the media object has a linked list of pointers to any cause object which causes it or is effected by it. Cause objects also hold two lists of pointers back to the media objects with which they are associated.

An example is seen in the last parameter shown in Figure 3. The cause object also includes metadata indicating the media element identifier which causes the effect.

Several different media objects might cause the same effect (e.g. a hero could die because he was poisoned, or because he was crushed to death) and there can be several different effects from a cause (e.g. because of the hero's death, the evil queen lived undefeated until the ripe old age of 103, and the broken-hearted princess made a vow of celibacy and became a nun). It is for this reason that a cause-effect relationship is represented using a cause object.

A second type of relationship that the editor may indicate is that of sequence. To indicate such a relationship, the editor arranges the media elements he wishes to group into the same rectangular area of the screen, ordering them from left-to-right in accordance with the sequence they should follow, clicks on a further button (not shown), which causes the cursor to change form and moves the cursor to a position to one corner of that rectangular area. Thereafter, the editor clicks the button on the mouse 22, and holds that button down whilst moving to an opposite corner of that rectangular area whereupon he releases the button. This results in a thick, solid rectangular line being drawn around the media elements contained within the rectangular area with a thick, short arrow being drawn in the top left-hand corner of the area defined by the rectangular line. In the example in Figure 4C, the editor has indicated that the media element representing Team B's fifteenth pass is followed by the media element representing Team B's sixteenth pass which is in turn followed by the media element representing Team B's first goal.

An editor might wish to indicate a sequential relationship of this nature where he feels that the media elements should be shown in the indicated order (if more than one of the media elements are selected by the template populator module). Media elements showing gardening at different times of year, for example, might be arranged into a sequence so that an element representing the garden in spring precedes an element representing the garden in summer and so on.

On creation of a sequence in this way, a sequence object is created in the object-oriented database as a container object containing the media objects associated with the media elements included within the sequence. As will be seen below, it is possible to generate a sequence which itself includes sequences. This hierarchical property is reflected in the first number in the identifier attributed to the sequence. Where the sequence includes only individual media elements, then the sequence identifier is of the form 1.x where x is simply incremented at each time a new sequence or group (explained below) at the first level of the hierarchy is formed. Hence the sequence shown in Figure 4C is given the identifier 1.1.

The media object (i.e. metadata) associated with each media element in the sequence has the position of the media element within that sequence added to it. The object-oriented database also records the fact that each media object is a child of (i.e. is included within) the newly created sequence object. An example of the sequence position metadata can be seen in the penultimate row of Figure 3.

In this embodiment, there is no metadata giving a description of groups or sequences, but there could be – such metadata might, for example, be entered by right clicking, selecting properties and entering text in the description field of the resulting dialog.

The third type of relationship an editor may wish to indicate between media elements is that of membership of a group. An editor might do this where he wishes to indicate that in a plurality of the media elements in the group are selected, then they should be shown together. A group is formed in the same way as a sequence, save for the order of the media elements on the screen being inconsequential and the editor clicking on a third button (not shown) representing a group prior to defining the rectangular area which contains the media elements to be included in the group.

This action creates a group object, a container object which contains the media objects associated with the media elements within the group. Group objects are also stored within the object-oriented database 54.

Figure 4D shows the display seen by a editor after he has formed media elements 0.13, 0.14 and 0.15 into a first-level group 1.2, 0.39, 0.40 and 0.41 into a second first-level sequence 1.3, media elements 0.52 and 0.53 into a second first-level group 1.4.

Figure 4E shows the editor joining the media element 0.51 and the third first-level sequence 1.4 into a second-level group 2.1. This creates another group object in the database, which is a container object for another group object. In this way a hierarchy of container objects can be built up in the object-oriented database 54.

Figure 4F shows the editor placing the two first-level sequences 1.1 and 1.3, the first-level group 1.2 and the second-level group 2.1 into a third-level group 3.1.

Figure 4G shows the editor defining a fourth-level sequence.

Figure 5 shows the relationships entered by the editor in hierarchical form and therefore reflects the object hierarchy which the editor has built up within the object-oriented database 54 on the hard disk 16.

Returning to Figure 2, the template creation tool 42 provides an interface for an editor and/or producer to specify the desired characteristics of a media article – thus creating a template object (Figure 6). The tool stores the metadata which constitutes this template object within the object-oriented database 54.

Like a media object, a template object for use in the present embodiment conforms to a comprehensive predefined data model. As can be seen from Figure 6, this predefined data model includes a title field, and a plurality of sections. Each section is a set comprising a name field, a query field, and, optionally, a constraint field. The template populator tool prompts the user to enter a name for the template and to indicate the section structure (top-level sections may themselves contain sections). The editor indicates the section structure using a graphical user interface component similar to the Folder List provided in Microsoft Windows Explorer for example. In the example given in Figure 6, the template has a flat structure of three sections.

Because the template encodes the media article characteristics using complex queries and a potentially deep structure, the editor interface divides the task of template creation between a plurality of roles. A person who is assigned an editor role defines the top-level structure of the template and a person who is assigned a producer role, (a producer normally having closer control of the actual product being created), refines the structure and adds queries (requests for information from the object-oriented database). In particular, as will be explained below, the producer specifies the linkages to the user profile thereby defining the 'balance of power' between themselves and the consumer.

The template creation tool provides an object browser which can be used to search for existing media objects and template objects. Existing templates can be modified and portions of a template can be copied into new templates.

Having defined the section structure using the graphical user interface mentioned above, perhaps using the media object browser, the editor / producer is provided with a graphical user interface which facilitates the process of query formation.

The editor uses this graphical user interface to enter query strings for each of the sections. The query string for the first section in Figure 6 indicates that candidate media objects to fill this slot in the template must have the string "Football Intro" in their URI.

The query string for a section can be considerably more complex, as is seen in the 'Main' section of the template of Figure 6. Here the query is written in the XPath language – the way in which that query is evaluated will be explained after the user profile metadata has been described below.

The editor also enters constraints for those sections where he wishes to place some constraint on the media elements represented by the media objects retrieved from the database in response to the query. Constraints are intended to restrict the way in which media objects are assembled by the template populator. Possible examples of constraints include time (e.g. this section must be 5 minutes long), space (e.g. this presentation must be viewed on a 640 * 480 pixel display), or number (there must be five news items in the 'headlines' section of a news programme).

The user profile creation tool 44 provides a user interface for generating user profiles like the user profile seen in Figure 7.

The user profile expresses the preferences of a particular user. As with media objects and the template objects, the data must adhere to a predetermined data structure or schema. The schema dictates that each user is identified by an identifier (the numeral '1' in this case). The 'Structural' element of the user profile indicates the things the user likes – in this case, the football team Team B, especially Paulo Di Canio and Trevor Sinclair, the Essex cricket team, especially Nasser Hussain, the actress Milla Jovovich and the actor Jimmy Nail.

The template populator program module (Figure 8) provides a process for the automatic assembly of an edit decision list in preparation for the synthesis of a set of media objects into a personalised media article for a consumer. On starting (step 60), the template populator takes as its inputs a specific template, a specific user profile, and an indicator of a store of media objects (in the present case, an indication of the location of the object-oriented database 54).

Once the specific template, user profile and store of media objects have been specified, the template populator examines (step 61) the template (Figure 6) for any queries written in the XPath language. If such a query is found, it is resolved using the Apache Project's Xalan processor to evaluate XPaths.

For example, the query reading:

```
( Team is "!profile(Profile/Sports//Team)") AND
(Action is "Goal"))
```

is resolved, when the user profile in Figure 7 (specified on starting the template populator program module) is considered, to:

```
(( Team is "Team B" OR "Essex" ) AND (Action is "Goal"))
```

The template populator then identifies the first section of the template (Figure 8) and iterates (steps 62 to 75) through the template's hierarchical structure.

Each iteration (steps 62 to 75) involves the next section in the template being found, any query in that section being executed (step 62) on the object-oriented database 54 to return a selection of relevant media objects.

The first iteration relates to the section named 'Intro' in Figure 6. The iteration begins (step 62) with the carrying out of the query contained with the section (i.e. URI contains "Football Intro"). Figure 9A illustrates the selection that results in this example.

Then, in step 64, a tree is constructed which includes the selected media objects as its 'leaves'. This construction takes place as follows: The parent object of the first selected media object is retrieved followed by its parent object and so forth until an object is reached which has no parent object associated with it (the 'Introduction' object in this example has no parent object, so is the only object included in the tree). At this point, a single linked list from the leaf object to the top-level container has been reconstructed. Another selected leaf object is examined (if more than one object is selected as a result of the query), and the ancestry of that leaf object is followed until either an object is retrieved that already exists in the linked list representing the ancestry of the first object or another top-level container is encountered. Repeating this process for all the other objects in the selection reconstructs the minimal tree containing those objects.

As indicated above, in the first iteration, the resultant tree contains only the 'Introduction' media object 0.1.

The subsequent steps, in the loop of instructions (steps 66 to 72) in the template populator program which alter the tree data structure which is stored and used in generating the edit decision list, have no effect in relation to the first section of the template, so will be described below in relation to the second iteration of the loop of instructions carried out on the second section of the template.

Throughout the iteration, the tree structure is stored in the PC's volatile memory 12.

At the end of each iteration of the group of instructions, a determination is made (step 74) as to whether the final section in the template has been considered. If not, the next section is identified (step 75) and next iteration carried out.

The second iteration is carried out in relation to the central section of the Football Goals Highlights template (Figure 6). In this case, the section query is carried out using the user profile (Figure 7) in order to resolve the query as described above to take account of the user's preferences.

The query (step 62) results in the selection of the media elements 0.12 and 0.53 (Figure 10A). A new tree data structure is created as described above and stores data representing the tree illustrated in Figure 10B.

Thereafter, the selection of media objects is expanded to take account of cause / effects relationships specified by the user (step 66). In detail, this step involves the examination of the metadata of each selected media object to find how many cause objects are associated with that media object. If no cause objects are found then the media object is moved to a list of resolved media objects. If only one cause object is found, then the cause object is moved to a list of cause objects, and the media object is moved to the list of resolved media objects. If more than one cause object is found, then each possible cause object is added to a list of possible cause objects (if it is not already present in that list) and the media object is added to a list of unresolved media objects.

In the present example, only one cause object is found (that illustrated in Figure 3B), so the media object 0.53 is moved to a list of resolved media objects (and added to the tree data structure associated with this iteration of the loop of instructions – as shown in Figure 10C) and the cause object is moved to a list of cause objects.

Where a list of possible cause objects is created, the cause object which causes the most unresolved media objects is found. This cause object is moved into the list of cause objects, and the media objects it causes are added to the resolved media object list mentioned above. This process is repeated until all the list of unresolved media objects is empty. In the present example, this step is not applied.

Each cause object in the list of cause objects is then examined to find how many media objects it was caused by. If only one media object causes it, then the cause object is moved to a list of resolved cause objects and that media object is added to the list of unresolved media objects. If more than one media object causes the cause object then those media objects are added to a list of possible media objects if they are not already present.

Since, in the present example, the only cause object (CO1) in the list of cause objects is caused by only one media object (0.52), then the cause object is moved to a list of resolved cause objects and the media object (0.52) is moved to the list of unresolved media objects.

Where a list of possible media objects is obtained, the media object which causes the most cause objects is found. The cause object is then moved into a list of resolved cause objects and the media object is added to the list of unresolved media objects. This is repeated until the list of cause objects is empty.

The above procedure is then repeated for any unresolved media objects (so that chains of causation are traced back to the original cause). In the present case therefore the above procedure is repeated for media element 0.52 and results in the addition of media object 0.51 to the tree associated with this iteration (Figure 10D).

The building of the tree (steps 62 to 66) is followed by sorting (steps 68 and 70) of the objects within the tree.

The first stage of sorting (step 68) takes account of the sequence information entered by the user. The is done by using the known 'Quicksort' algorithm to place the nodes of the tree in the correct order as identified by the sequence position metadata associated with the object. This is done starting at the top of the tree and then moving towards the leaves (i.e. the media objects) of the tree.

The second stage of sorting takes account of cause / effect linkages between the members of a group or the descendants (i.e. objects further down the tree) of the members of a group. Where groups do not have such cause / effect linkages then this stage of sorting need not be carried out on those groups.

The second stage of sorting begins by labelling each member of a group in the tree with all the causes and effects attached to it (if it is a media object) or to any of its descendants (if it is a container).

Further labels are then added to the object metadata to reflect the logical relation that if a causes b and b causes c, then a causes c. The same is done to reflect the logical relation that if f is caused by e, and e is caused by d, then f is caused by d.

The known Quicksort algorithm is then used to ensure that causes are shown before effects. As those skilled in the art will know, implementations of Quicksort allow the user to define a function which gives the order of two objects passed to it. In the present case, a function is provided which indicates that a goes before b is a causes b and that d comes after c, if d is caused by c.

Thus, at the end of the sorting steps (steps 68 and 70) in the second iteration, media elements 0.12, 0.51, 0.52 and 0.53 form the leaf nodes of the tree associated with the central section defined in the Football Goal Highlights template (Figure 6).

The template populator then evaluates any constraints and updates the tree accordingly (step 72). To evaluate a time constraint, the duration of each media object included within the tree is calculated by subtracting the 'Out' property from the 'In' property, and these durations are added together to reach an actual duration. If this duration is found to be greater than the target duration, then media objects are removed from the tree. If this duration is less than the target duration, then media objects are added to the tree.

In the present embodiment this pruning or growing of the tree is done in accordance with the Interest Value metadata associated with the media objects.

Where the actual duration is longer than the target duration, the following process is carried out:

1) If the difference between the target duration and the actual duration is less than the duration of the shortest media element in the tree, then the process terminates;

2) Otherwise, a list of the media objects within the tree is created and this list is sorted in order of the 'Interest Value' property.

3) The media object with the lowest value is removed (assuming that this object is not the cause of one or more of the other media objects in the list) from the tree and the actual duration re-calculated. The difference between the target duration and the actual duration is re-calculated and the above steps repeated until the difference is less than the duration of the shortest media element remaining within the tree.

Where the actual duration is less than the target duration, the following process is carried out:

A) The query in the section is amended by removing its last 'AND' operator and the condition which follows it, or if there is no 'AND' operator, by adding an 'OR' operator followed by a condition such as 'Interest Value' > 0.6. A new tree for the current section is then created in the same way as the original tree. This process is repeated until the actual duration is greater than the target duration. Thereafter, steps 1) to 3) above are carried out.

Once this pruning or growth has been carried out, the second iteration ends.

It will be clear that the third iteration will merely generate a tree comprising media object 0.99.

When all sections have been populated with media object metadata and sequenced in accordance with the queries, constraints and user preferences provided, the template populator outputs (step 78) the edit decision list (Figure 11) by concatenating the media elements found at the leaves of the trees generated in the three iterations of the loop.

The edit decision list (Figure 11) produced by the template populator program module (46) is passed to the content synthesiser module (48). In the present example, the content synthesiser module outputs one scene after another in a streamed video presentation or concatenates the scenes together in order to produce a programme file.

The content synthesiser provides a process to automatically synthesise a set of media elements into a personalised media article for a consumer. The synthesiser is similar to a conventional non-linear editing system in that it uses the edit decision list (Figure 11) to locate and assemble a set of media elements into a completed media article. However, while a conventional editing tool is used only during post-production and resides on the editor's premises, the content synthesiser is a portable, lightweight application which can be deployed at the edge of the network, or actually within the consumer's device (such as a PC or personal digital recorder). This means that media articles can be synthesised at the point of delivery, with considerable benefits in terms of the level of personalisation which can be achieved. Furthermore, changes in any metadata including template, media object or user profile can be reflected dynamically in the resulting media article.

When invoked by the consumer, the content synthesiser causes a user profile and template to be passed to the template populator, which processes these as described above returning an edit decision list to the synthesiser. The edit decision list is then parsed to discover which media elements are required and how they interact with each other. A timeline comprising various references to various media elements within the content store 50 is then assembled according to this information. Transition effects (examples of which are defined in the ANSI/SMPTE 258M/1993 standard) are applied to the media where required. Finally, any caption text and graphical overlays are parsed and rendered to a static image, which is then added to the timeline with a high priority ensuring they are visible above any other graphical elements.

Once the timeline has been assembled, the personalised presentation is rendered in real time using a suitable media compositor technology, such as Apple Quicktime or Microsoft DirectShow. The modular nature of the compositor means that a content synthesiser could be more easily implemented within an embedded architecture such as a set-top box.

The graphical user interface provided by a preferred embodiment of the invention will now be described with reference to Figures 11 to 20.

Figure 12 shows an application window which appears when a media article application program is run on the computer of Figure 1. It will be seen that provides many well-known graphical user interface components.

Figure 13 shows the display after the user has selected a 'project' by using the File menu or 'Open' button in order to select a project to open. In this preferred embodiment, the directory structure in the content store 50 is organised into project directories at the highest level, each of which contains one or more 'media bins' – i.e. directories holding a collection of media elements, and one or more 'template bins' – i.e. directories holding a collection of templates. It will be seen in Figure 13 that, on the user opening the 'Paper Mill 03.db' project, a project sub-window is displayed showing that the 'Paper Mill 03.db' project directory contains two media bins called 'Paper Media' and 'juliamark' and one template bin called 'Paper Templates'.

Figure 14 shows an additional sub-window which opens on the user double-clicking on the 'Paper Media' media bin. The sub-window displays a plurality of rows, each comprising a number of column entries. As can be seen, by moving the mouse pointer over the title bar of the sub-window, a list of all the attributes included in the structured data model for describing media elements is given to the user. By clicking with the mouse, a user is able to select which elements within that structured data model are displayed in the 'Paper Media' media-bin sub-window. It will be noted that the structured data model corresponds closely to the structured data model which underlies the metadata shown in Figure 3.

On double-clicking on one of the thumbnails provided in the first column of each row, the metadata associated with the media element that the thumbnail represents is shown in a further sub-window (Figure 15). The value given to each attribute is editable. In addition, some of the attributes have vocabularies associated with them (i.e. a limited number of predetermined values for the attribute). Where this is the case, right-clicking on the value field will cause a drop-down list to be displayed containing those predetermined values. A user can select one of those predetermined values from that list in the normal manner.

When a user selects a plurality (and possibly all) the media elements included in the sub-window illustrated in Figure 14, right-clicks and selects the 'Relationships' option, a further sub-window is added to the display (Figure 16) which shows the relationships between the various media elements included within the selection. In a manner similar to that shown in Figures 4A to 4G, groups of media elements are illustrated by enclosing the corresponding thumbnails in a solid border, sequences are similarly illustrated – save for the border including an arrow symbol near the top left-hand corner. Also illustrated are two diamonds representing cause objects as described in relation to the first embodiment. The left-hand cause object shows that a cause/effect relationship exists between media element JKY 08a and JKY 08b.

In a refinement of the first embodiment described above, the user can right-click on a diamond shape representing a cause object and is then provided with a list of options – namely, 'Backward', 'Forward', 'Bi-Directional', 'Detach All' and 'Delete'. The first three refer to the manner in which the tree expansion process described in relation to the first embodiment above expands the tree when it selects a media element which has a cause object attached to it. If 'Forward' is chosen, then the selection of JKY 08a will result in JKY 08b being added to the tree, but not vice-versa. 'Backward' has the opposite effect – namely, if JKY 08b is selected then JKY 08a will be added to the tree, but not vice-versa. If 'Bi-directional' is selected then selection of either one will result in the tree expansion adding in the other. The choice of 'Forward', 'Backward', and 'Bi-Directional' is stored as an additional row of metadata in Figure 3B.

Figure 17 shows a 'Vocabularies' sub-window which the user can open by selecting a 'Vocabularies' option from the Edit menu. The underlying structured data model for a media object is seen, once again, in the top-left box of the sub-window. The user may define vocabularies – i.e. lists of acceptable values for any attribute he selected in that data model. In Figure 17, the user has selected the attribute 'Place' and has defined the values 'By the canal', 'Paper Mill' and 'Sainsbury' as possible values of that attribute – this vocabulary is associated with the currently selected project and is stored in the object-oriented database (Figure 2 – 54).

Figure 17 also illustrates the extensibility of the structured data model. This extensibility is enabled by allowing the user to add domain specific attributes (it will be remembered that examples of such attributes are included also in Figure 3). The domain specific attributes are shown in the bottom left-hand box of the sub-window illustrated in Figure 17, as are an 'Add' and 'Del' buttons which may be used to add or delete domain specific attributes. The domain specific attributes are added to the data model associated with the project which is currently open.

Figure 18 illustrates the steps a user follows in order to generate template metadata akin to that shown in Figure 6. As with the generation of media object metadata the user first opens a project sub-window. Thereafter, the user double clicks on one of the displayed template bins (in the example shown the user has double-clicked on the 'Paper templates' template bin) to bring up a 'Query Builder' sub-window. The tree structure in the left-hand side of that sub-window can be seen to be similar to the table in Figure 6, with each '< New Scene >' corresponding to a section.

The circles in the tree structure represent filters which select media objects from the media bin. There are three types of filters as follows:

i) Circles containing '=, or >, or < , or ≠'
these filters select, for a user-selected attribute, all the media objects in the media bin which have a value for that attribute that equals the value shown alongside the circle, or is greater than that value etc.

ii) Circles containing (*) which represent filters which select all the media objects in the media bin which contain the string shown alongside the circle.

The logical operators used to combine the results of the filtering operations are known as combiners and are displayed as triangles. There are three types of combiners called 'random', 'sequential', and 'Either/Or'.

The sequential combiner merely displays the results of the filtering operations directly below it in the tree in the sequence in which they are placed in the tree. Each sequential combiner has a straight arrow through it – examples are seen in the first and third sections of the 'Machine (860)' template seen in Figure 18.

The random combiner displays the results of the one or more filters underneath it in the tree in a random order. A random combiner has a zigzag arrow through it. Examples are seen in the second section of the Machine (860) template seen in Figure 18.

The either/or combiner (not illustrated) chooses the results of the two filter branches beneath it. An either/or combiner has a split arrow through it.

Figure 19 shows how a user can amend the tree and thus create template data similar to that shown in Figure 7. When a filter is added the user is given a choice of attributes and then can enter the desired value.

Another element which can be added to a tree is a 'funnel'. This acts like a filter, allowing only a user-defined number of randomly-chosen media objects up to the higher stage in the tree.

Where filters are nested, each is applied in turn to the results of the filter lower down the tree.

In order to generate an edit decision list, the user clicks on the button marked '!' in the toolbar whilst a template is selected. The edit decision list is then added to an edit decision list history sub-window displayed when the appropriate button on the toolbar is pressed. Right clicking on an edit decision list in that window gives the user the options of playing a video in accordance with that edit decision list, previewing the storyboard (the result of which is shown in Figure 20), importing or exporting the edit decision list, editing the edit decision list in its textual form or deleting the edit decision list.

An additional feature is provided in a refinement of the preferred embodiment of the present invention. The Graphical User Interface offering this feature to the user is illustrated in Figure 21. On selecting the words 'Jacky, Problem', and inserting a filter, the user is presented with a check-box. On ticking that check-box, the user is provider with a list of drop-down controllers as shown. By selecting one of those controllers, the user is able to apply a 'controller' to that filter. The controller can be provided more than once (it will be seen that the first controller is applied twice in the template illustrated in Figure 21).

Figure 22 shows a pair of slider controls can be moved by the user. By dragging, for example, the uppermost slider to a position further to the left, the 'Conceptual_PlotValue' associated with controller #1 can be reduced (and vice-versa). By altering the position of the slider, the user is able to change the template generated by the Query Builder directly from the media article preview window (Figure 22), rather than having to re-define the filter(s). This not only provides an easier method of updating the filters within the template but also forms the basis for a simplified user interface that does not require access to the query builder and the like. In other words, a non-technical user is provided with a means for generating bespoke video presentations.

The present invention may be embodied in many different ways. For example, the embodiments described above may be altered in one or more of the ways listed below to provide alternative embodiments of the present invention (this list is by no means exhaustive):

I) in the above embodiment the input program modules generated data structures having a predetermined structure. In some embodiments, the predetermined structure is provided by a document conforming to the XML Schema guidelines;

ii) the constraints section in the template might be variable by the user – for example, a user could be provided with a graphical user interface in which he can select the duration of the media article he wishes to see. A corresponding value can then be added to the template object by the template populator program;

iii) one constraint which might be added to each selection or combination of selections could be a constraint on the number of times a given media element appears in the finished media article. It might, for example, be dictated that any media element should only appear once in a given media article;